

```
import numpy as np
# import library numpy ย่อชื่อเป็น np
```

```
data1 = [1, 2, 3, 4, 5]
# [1, 2, 3, 4, 5] -> data1
```

```
arr1 = np.array(data1)
# [1, 2, 3, 4, 5] -> arr1
```

```
data2 = [range(1, 5), range(5, 9)]
# 1 ถึง 4, 5 ถึง 9 -> data2
```

```
arr2 = np.array(data2)
# [1, 2, 3, 4], [5, 6, 7, 8] -> arr2
```

```
arr2.tolist()
# แปลง arr2 เป็น list
```

```
→ [[1, 2, 3, 4], [5, 6, 7, 8]]
```

```
np.zeros(10)
# สร้าง array ที่มีค่าเป็น 0 จำนวน 10 ตัว
```

```
→ array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

```
np.zeros((3, 6))
# สร้าง array ที่มีค่าเป็น 0 จำนวน 3 แถว 6 หลัก
```

```
→ array([[0., 0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0., 0.],
         [0., 0., 0., 0., 0., 0.]])
```

```
np.ones(10)
# สร้าง array ที่มีค่าเป็น 1 จำนวน 10 ตัว
```

```
→ array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

```
np.linspace(0, 1, 5)
# สร้าง array ที่มีค่า 0 ถึง 1 ให้มี 5 ตัว โดยจะแบ่งเว้นระยะแต่ละตัวให้เท่าๆ กัน
```

```
→ array([0. , 0.25, 0.5 , 0.75, 1.  ])
```

```
np.logspace(0, 3, 4)
# สร้าง array ที่มีค่า  $10^0$  ถึง  $10^3$  ให้มี 4 ตัว โดยจะแบ่งเว้นระยะแต่ละตัวให้เท่าๆ กัน
```

```
→ array([ 1., 10., 100., 1000.])
```

```
int_array = np.arange(5)
# สร้าง array ที่มีค่า 0 ถึง 4
```

```
float_array = int_array.astype(float)
# แปลง int_array เป็น float_array
```

```
arr1.dtype
# แสดง dtype ของ arr1
```

```
→ dtype('int64')
```

```
arr2.dtype
# แสดง dtype ของ arr2
```

```
→ dtype('int64')
```

```
arr2.ndim
# แสดง dimension ของ arr2
```

```
→ 2
```

```
arr2.shape
# แสดง shape ของ arr2
```

 (2, 4)

```
arr2.size  
# แสดง size ของ arr2
```

 8

```
len(arr2)  
# แสดงว่าใน arr2 มีกี่ตัว
```


 2

```
arr = np.arange(10, dtype=float).reshape((2, 5))  
# สร้าง array ที่มีค่า 0 ถึง 9 โดย dtype เป็น float และ reshape เป็น 2 แถว 5 หลัก
```

```
print(arr.shape)
```

 (2, 5)

```
print(arr.reshape(5, 2))
```



```
[[0. 1.]  
 [2. 3.]  
 [4. 5.]  
 [6. 7.]  
 [8. 9.]]
```

```
a = np.array([0, 1])  
# [0, 1] -> a
```

```
a_col = a[:, np.newaxis]  
# [0, 1] -> a_col โดยเรียงใหม่เป็น 2 มิติ
```

```
print(a_col)
```




```
[[0]  
 [1]]
```

```
a_col = a[:, None]  
# [[0]  
# [1]] -> a_col
```

### #Stack array

```
a = np.array([0, 1]) # [0, 1] -> a  
b = np.array([2, 3]) # [2, 3] -> b  
ab = np.stack((a, b)).T # [[0, 2], [1, 3]] -> ab  
print(ab)  
np.hstack((a[:, None], b[:, None])) # สร้างแกนใหม่
```



```
[[0 2]  
 [1 3]]  
array([[0, 2],  
       [1, 3]])
```

### #Selection

```
arr = np.arange(10, dtype=float).reshape((2, 5))  
arr[0] # 0th element (slices like a list)
```

 array([0., 1., 2., 3., 4.])

```
arr[0, 3] # row 0, column 3: returns 4
```

 3.0

```
arr[0][3] #
```

 3.0

### #Slicing

```
arr[0, :] # row 0
```

```
↔ array([0., 1., 2., 3., 4.])
```

```
arr[:, 0] # column 0
```

```
↔ array([0., 5.])
```

```
arr[:, :2] # first two columns
```

```
↔ array([[0., 1.],
        [5., 6.]])
```

```
arr[:, 2:] # last two columns
```

```
↔ array([[2., 3., 4.],
        [7., 8., 9.]])
```

```
arr2 = arr[:, 1:4] # column 1 to 3
print(arr2)
```

```
↔ [[1. 2. 3.]
    [6. 7. 8.]]
```

### #Vectorized operations

```
nums = np.arange(5) # [0, 1, 2, 3, 4] -> nums
```

```
nums * 10 # ([0, 1, 2, 3, 4])*10
```

```
↔ array([ 0, 10, 20, 30, 40])
```

```
nums = np.sqrt(nums) # sqrt([0, 1, 2, 3, 4]) -> nums
```

```
np.ceil(nums) # บัดเศษ
```

```
↔ array([0., 1., 2., 2., 2.])
```

```
np.isnan(nums) # ไม่ได้เป็นเลขหรือไม่
```

```
↔ array([False, False, False, False, False])
```

```
nums + np.arange(5) # [0, 1, 2, 3, 4] + [0, 1, 2, 3, 4]
```

```
↔ array([0.         , 2.         , 3.41421356, 4.73205081, 6.         ])
```

```
np.maximum(nums, np.array([1, -2, 3, -4, 5])) # max(nums, [1, -2, 3, -4, 5])
```

```
↔ array([1.         , 1.         , 3.         , 1.73205081, 5.         ])
```

### #Compute Euclidean distance between 2 vectors

```
vec1 = np.random.randn(10)
```

```
vec2 = np.random.randn(10) # สร้าง 2 vector ที่มีค่า random
```

```
dist = np.sqrt(np.sum((vec1 - vec2) ** 2)) # sqrt(sum((vec1 - vec2) ** 2))
print(dist)
```

```
↔ 4.049900449769588
```

### # math and stats

```
rnd = np.random.randn(4, 2) # random normals in 4x2 array
```

```
rnd.mean() # average
```

```
↔ 0.3891968446336621
```

```
rnd.std() # standard deviation
```

```
↔ 1.2871831703874408
```

```
rnd.argmax() # index of min
```

```
↵ 1
```

```
rnd.sum() # sum
```

```
↵ 3.113574757069297
```

```
rnd.sum(axis=0) # sum across columns
```

```
↵ array([1.60412655, 1.50944821])
```

```
rnd.sum(axis=1) # sum across rows
```

```
↵ array([-0.73279757, 1.73250096, 1.74835547, 0.3655159 ])
```

### # methods for boolean arrays

```
(rnd > 0).sum() # number of positive values
```

```
↵ 5
```

```
(rnd > 0).any() # any positive values?
```

```
↵ True
```

```
(rnd > 0).all() # all positive values?
```

```
↵ False
```

### # random numbers

```
np.random.seed(12234) # set seed
```

```
np.random.rand(2, 3) # random numbers in [0, 1]
```

```
↵ array([[0.00630595, 0.20303476, 0.76478993],  
        [0.55513384, 0.74358546, 0.93777808]])
```

```
np.random.randn(10) # random numbers from standard normal
```

```
↵ array([-2.79962074e-01, 1.31281104e+00, -9.27155784e-01, -4.01302169e-01,  
        -2.31085929e+00, -2.08460156e+00, 4.59241643e-01, 1.62191344e+00,  
        1.94515120e-01, -2.08631547e-03])
```

```
np.random.randint(0, 2, 10) # random integers in [0, 2]
```

```
↵ array([0, 0, 0, 1, 1, 0, 1, 1, 1, 1])
```

### #Broadcasting

```
a = np.array([[ 0, 0, 0],  
             [10, 10, 10],  
             [20, 20, 20],  
             [30, 30, 30]])  
b = np.array([0, 1, 2])  
print(a + b)
```

```
↵ [[ 0  1  2]  
   [10 11 12]  
   [20 21 22]  
   [30 31 32]]
```

